CHICAGO TIMES

THE NEWSLETTER OF THE CHICAGO AREA TI-99/4(A) USER'S GROUP, VOLUME 4 #6

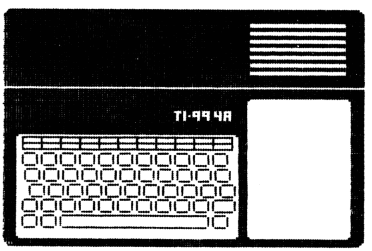
THIS IS YOUR LAST NEWSLETTER
IF YOU HAVE NOT RENEWED

EDITOR: Carole Goldstein

31 JAN. 1984 1985

The Chicago TImes is published 10 TImes a year, monthly, except during June and July. Chicago TImes is not affiliated with Texas Instruments in any way and is supported only by its subscribers and advertisers. Subscriptions are free with membership in the Chicago TI99/4A User's Group. The Chicago TImes is also distributed free to any other User's Group that wishes to reciprocate. Articles contained herein may be re-printed by another User's Group Publication provided credit is given to the Chicago TImes as the original source of the article. Comments and letters are welcome, as is the submission of original articles and programs.

THE TI-99/4A COMPUTER



THE FEB MEETING

will be held on Saturday, Feb. 9, 1984, from 1:00 to 3:00 PM in the Fireside Lounge at Triton College. At this meeting we will have a demonstration by Harry Goodman of the RB Robot Corporation. He will be showing us his RB5X robot which can be programed through the RS232 port of our TI 99/4A computers.

IN THIS ISSUE...

THE DISASSEMBLY	Dave Wakely	Page 3
PRACTICAL PROGRAMMING PRACTICES		Page 5
BASICALLY YOURS	Rich Klein	Page 9
TRADING TImes		Page 11
FORTH CHAPTER	Seaborn Smith	Page 12
966-2342	Irwin Goldstein	Page 13
VIDEO MODES	Jim Ness	Page 14
LEARNING ASSEMBLY	Nick Iacovelli Jr.	Page 15
FIFTH OF FORTH	Stephan Meyers	Page 18
USING TI-WRITER	Roger Helm	Page 22
REMarks	Carole Goldstein	Page 23

BULLETINS:

The Business S.I.G will meet after the regular meeting this month.

Congratulations to Sam Pincus on his election to President of this group!

The User's Group Program Library is now on line on the group bulletin board

The basic class held by Sam Pincus will continue after the meeting. Please bring your consoles and monitors. Class is still open to all members at no charge.

Don't forget to bring your entries for the programming contest starting at the February meeting.

Program listings in the PRACTICAL PROGRAMMING PRACTICES section of this newsletter are printed exactly as they will appear on your screen as you key them in.

THE DISASSEMBLY: Dave Wakely

With the new year has come a new order of things for the group. For those who could not be at the January meeting and participate in democracy in action, Sam Pincus was elected president of the group for 1985. He survived a closely contested race over Stephan Meyers, who was subsequently appointed to the executive board as a member at large. Sam ran on a platform of fiscal responsibility encompassing reductions in the federal budget deficit and an equitable tax structure, but we elected him anyway. If the government's "Star Wars" plan goes through, word also has it that Sam may be doing some Extended BASIC programming for the Pentagon. I hear he has some great ideas on the use of sprites and the nuclear umbrella. This is Sam's second term in office, as he followed group founder Jerry Strauss into the president's role in 1983. Also, volunteers Jon and Sandy Bartels have taken over as program coordinators, in which capacity they will be responsible for lining up the presentations at the meetings for this year.

The January meeting was a "do it ourselves" presentation, as group members Bill Hoffman, Bob Lee, Rich Klein and Ken Czerwinski gang-demoed just about all the computer printer technology now available. A surprise was the $3,500 laser printer that Ken Czerwinski brought along. Lest you think that Ken was the winner of one of the weekly Illinois lotto drawings, it should be pointed out that the Hewlett-Packard laser printer belongs to Triton College, where Ken is the Assistant Director of Programming. The other presenters had to lug their equipment from their homes, and the group thanks them for the fine show. Personally, I thought the experiment with the camera went much better than I had anticipated, as Murphy's law was held in temporary abeyance.

The meeting ended with what appeared to be a large mass of people up at the front of the room attempting to learn BASIC, which from the looks of the faces in the crowd was a less intimidating experience than trying to learn it alone. Closer inspection revealed this to be part one of the three part course taught by Sam Pincus, who stated that the size of the class surprised him. It did not surprise me, perhaps because only I knew how many people came up to me last year and asked when the group would help them learn programming. Sam also stated he was pleased with the outcome, noting that he was able to cover more material than he usually does when teaching for TI. No doubt this is a reflection on the quality of our membership. Parts two and three of the course will occur after the main meetings in February and March.

To paraphrase someone or other, "Do you like the "new" group electronic bulletin board, or what?" I couldn't begin to guess the number of hours of work that have been put in by Sysops Butch and Carole, not to mention the various programmers who have been working on the board. I am told that the evolution will continue, and, if I haven't used up my quota of cliches for this column yet, I am also told that some of the upcoming changes will knock your socks off. There must be some good stuff going on there because I can never seem to get on the board anymore.

No sooner had I mentioned the few shortcomings of PFKEYS, the program I reviewed last month, than author Jim Kryzak informed me at the meeting that he is working on corrections for many of the points I noted. It appears there will someday be a PFKEYS II after all, and advance word

is that while it will cost slightly more, it will feature a fast XB loader and at least some of the features I listed. Jim even provided a patched-up PFKEYS with some coding to help with a dump to my Okidata printer, but alas, I haven't had time to try it out yet. How come I never got this kind of service from TI?

Lest anyone become suspicious of my generally favorable opinions; yes, I paid for PFKEYS, and no; I have no financial stake in this or any other product I might review. Nor are my reviews always positive. Some may remember the general drubbing I gave to the SST Compiler last year. Later, I somehow talked myself into purchasing the SST Expanded BASIC compiler, expecting better, and in a further moment of madness also purchased the Advanced Software BASIC compiler. To complete the lunacy, I bit at a review in MICROpendium and ordered Compiler 99 from Amerisoft in Georgia, advertised as a program that requires absolutely no rewriting of any TI BASIC code before compiling. It is now closing in on six weeks since my check went out, and no Compiler 99 has appeared. Sigh. Why so many? Am I a sucker for a good compiler? Yes, and the problem is that a good one has yet to appear. When the folks in Georgia get their act together (if they aren't, at this moment, somewhere in Brasil with my money) and send me Compiler 99 I will do a comparative review of the currently known TI compiler market.

Other random TI stuff: Stephan Meyers, in addition to running the FORTH SIG and running for group President, has agreed to rail loudly and wax eloquently on behalf of the 99/4A to the folks at Keyfax, the Chicago-area graphics-based, interactive video information service. I expect a full report soon...SUBFILE99, the on-again, off-again TI magazine on the SOURCE, is once again off. No explanations from Mike Amundsen, the SF99 owner/operator...Is everyone else waiting around for the January MICROpendium, or did I forget to send my renewal in?...I refuse to take any cheap shots at the publication schedules of either HCM or the Smart Programmer, especially since everything possible has already been said...There is no truth to the rumor that TI FORTH afficionado Hank Ellermann is legally changing his name to Ellermann Hank to conform to FORTH's reverse polish notation. When you think about it though, group member Ron Binkowski would be a better candidate for the switch...

For those who might be wondering, my official title with the group is now either vice president or ex-president, depending on who you talk to. I applied for the position of Group Gnome, but was informed that someone named Grant Schmalgemeier has been performing adequately in that capacity. Speaking of titles, shortly after I joined this group in 1981 and volunteered to start a newsletter, the executive committee consisted of myself, president Jerry Strauss, and librarian George Holod. It was at the last meeting that I was informed by George that he has received his degree in Chemical Engineering from IIT and taken a job in California. In addition, George managed his own TI software company, Data Systems, which he states he is now disbanding, although he may reestablish it when he gets to the land of the granola worshippers. George, good luck, we'll miss you, and remember what Mark Twain once said, "The U.S. is actually tilted toward the southwest, and everything loose rolled to California!"

Happy computing.

PRACTICAL PROGRAMMING PRACTICES:

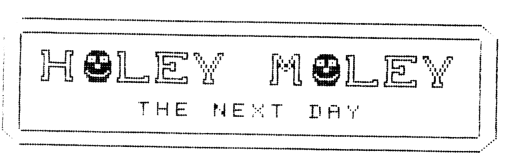
The following program was written in basic by David Punke. Use it to test your ability to change decimal numbers into Roman Numerals.

```
90 REM BY DAVID PUNKE
100 CALL CLEAR
110 RANDOMIZE
120 X=INT(3000*RND)+1
130 PRINT "DECIMAL NUMBER ="
;X
140 X$=""
150 X=INT(X)
160 IF X1000 THEN 200
170 X=X-1000
180 X$=X$&"M"
190 GOTO 160
200 IF X900 THEN 230
210 X=X-900
220 X$=X$&"CM"
230 IF X500 THEN 260
240 X=X-500
250 X$=X$&"D"
260 IF X400 THEN 290
270 X=X-400
280 X$=X$&"CD"
290 IF X100 THEN 330
300 X=X-100
310 X$=X$&"C"
320 GOTO 290
330 IF X90 THEN 360
340 X=X-90
350 X$=X$&"XC"
360 IF X50 THEN 390
370 X=X-50
380 X$=X$&"L"
390 IF X40 THEN 420
400 X=X-40
410 X$=X$&"XL"
420 IF X10 THEN 460
430 X=X-10
440 X$=X$&"X"
450 GOTO 420
460 IF X9 THEN 490
470 X=X-9
480 X$=X$&"IX"
490 IF X5 THEN 520
500 X=X-5
510 X$=X$&"V"
520 IF X4 THEN 550
530 X=X-4
540 X$=X$&"IV"
550 IF X1 THEN 590
560 X=X-1
570 X$=X$&"I"
580 GOTO 550
590 INPUT "ROMAN NUMERAL ?
":A$
600 IF X$=A$ THEN 640
610 PRINT "YOU'RE WRONG!!!!"
620 GOTO 590
630 PRINT
640 PRINT "YOU'RE RIGHT!!!!"
650 PRINT
660 GOTO 120
```

2021 correction: Several lines above are in the form if X40 then 420 - every time you see an X followed by a number, please insert a less than so that eg line 550 becomes IF X < 1 THEN 590.

\- - - - - - - - - - - - -

The following is a sequel to John Behnke's popular Holey Moley game which is in the downloading section of the group BBS and also available through the library. In Holey Moley The Next Day, the moles have gotten away and invaded your carrot patch. you must maneuver around with your joystick to take care of them before they eat your carrots.

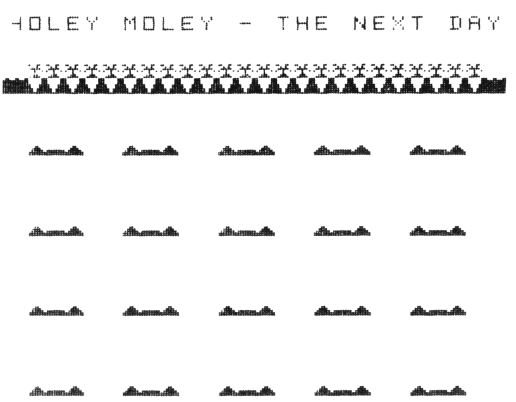


```
100 !HOLEY MOLEY II
110 !BY John Behnke 01/85
120 !TI X-BASIC
130 CALL CLEAR :: CALL SCREE
N(2)
140 CALL PEEK(-28672,SP):: I
F SP=0 THEN 180
150 CALL SPGET("GOT",Y$,"ONE
",Z$)
160 Y$=SEG$(Y$,1,35):: Z$=SE
G$(Z$,1,45)
170 CALL SPGET("HOW",LA$)::
LA$=SEG$(LA$,1,45)
180 RANDOMIZE
190 CALL SCREEN(2)
200 DIM CC$(24)
210 FOR I=1 TO 24 :: READ CC
$(I):: NEXT I
220 DATA ONE,TWO,THREE,FOUR,
FIVE,SIX,SEVEN,EIGHT,NINE,TE
N
230 DATA ELEVEN,TWELVE,THIRT
EEN,FOURTEEN,FIFTEEN,SIX TEE
N,SEVEN TEEN,EIGHT TEEN
240 DATA NINE TEEN,TWENTY,TW
ENTY ONE,TWENTY TWO,TWENTY T
HREE,TWENTY FOUR
250 CALL SPGET("I",O$,"GOT",
P$,"A",Q$,"PROBLEM",R$)
260 FOR I=0 TO 14
270 CALL COLOR(I,16,1)
280 NEXT I
290 CALL CHAR(96,"00CC287C92
10387C")
300 CALL COLOR(9,4,1)
310 CALL CHAR(104,"838383C7C
7EFEFFF")
320 CALL COLOR(10,8,10)
330 CALL CHAR(128,"0",129,"0
2071F0507070307",130,"0",131
,"40E0F8A0E0E0C0E0")
340 CALL CHAR(132,"0F1F1F1D1
D180E0F0E0E0E0E0C0C3C3C")
350 CALL CHAR(134,"F0F8F8B8B
81870F0707070303033C3C")
360 CALL CHAR(136,"070F1F111
5313F3E3E3F2F37181F0F0F")
370 CALL CHAR(138,"E0F0F888A
88CFC7C7CFCF4EC18F8F0F0")
380 CALL CHAR(140,"0000183C7
EFFFF0000000000FFFFFF0099FFF
FFFFFFFFFF")
390 CALL COLOR(14,8,1)
400 GOSUB 1370
410 CALL CLEAR
420 GOSUB 1710
430 DISPLAY AT(1,1):" HOLEY
MOLEY - THE NEXT DAY"
440 CALL HCHAR(5,1,142,32)
450 CALL HCHAR(5,5,104,24)
460 CALL HCHAR(4,5,96,24)
470 FOR I=9 TO 24 STEP 5
480 FOR J=5 TO 25 STEP 5
490 CALL HCHAR(I,J,140)
500 CALL HCHAR(I,J+1,141)
510 CALL HCHAR(I,J+2,140)
520 NEXT J
530 FOR K=0 TO 3 :: CALL SPR
ITE(#(I-4)/5*4-3+K,60,1,I*8-
2,200):: NEXT K
540 NEXT I
550 DIM R(4,5),C(4,5)
560 FOR I=1 TO 4
570 FOR J=1 TO 5
580 R(I,J)=I+29 :: C(I,J)
=J-3
590 NEXT J :: NEXT I
600 CARROTS=24
610 ROUND=1
620 TIME=24
630 SKIP=1
640 RANDOMIZE
650 CALL MAGNIFY(3)
660 CALL SPRITE(#18,128,6,38
,117,#19,132,6,54,117)
670 ROW=38 :: COL=117
680 CALL TALK("#WHAT WAS THA
T# OH NO")
690 IF SP>0 THEN CALL SAY(,O
$,,P$,,Q$,,R$)
700 CALL TALK("#READY TO STA
RT#")
710 DISPLAY AT(3,1):" PRESS
FIRE BUTTON TO START"
720 CALL KEY(1,K,S):: IF S=0
 THEN 720
730 DISPLAY AT(3,1):""
740 FOR I=1 TO 10+RND*ROUND
750 RR=INT(4*RND)+1
760 CC=INT(5*RND)+1
770 DD=INT(10*RND)+1
780 IF DD<3 THEN TTT=3 ELSE
TTT=TIME
790 CALL SPRITE(#20,136,10,R
(RR,CC),C(RR,CC))
800 FOR K=R(RR,CC)TO R(RR,CC
)-16 STEP -SKIP
810 CALL LOCATE(#20,K,C(RR,C
C))
820 NEXT K
830 FOR L=1 TO TTT
840 CALL JOYST(1,X,Y)
850 IF Y=0 AND X=0 THEN 920
860 X=X
870 Y=-Y
```

```
880 IF ROW+Y<1 OR ROW+Y>190
OR COL+X<1 OR COL+X>235 THEN
 1100
890 ROW=ROW+Y :: COL=COL+X
900 CALL SOUND(1,-3,0)
910 CALL LOCATE(#18,ROW,COL,
#19,ROW+16,COL)
920 CALL KEY(1,K,S):: IF S=0
 THEN 1100
930 CALL SOUND(50,-5,0)
940 CALL LOCATE(#18,ROW+8,CO
L)
950 CALL SOUND(50,-1,0)
960 CALL LOCATE(#18,ROW,COL)
970 CALL COINC(#19,#20,16,HI
T)
980 IF HIT THEN 990 ELSE 102
0
990 CALL DELSPRITE(#20)
1000 IF SP>0 THEN CALL SAY(,
Y$,,Z$)ELSE CALL SOUND(100,-
7,0)
1010 GOTO 1140
1020 CALL SOUND(150,110,0)::
 GOTO 1110
1030 CALL DELSPRITE(#20)
1040 CARROTS=CARROTS-1
1050 CALL HCHAR(4,CARROTS+5,
32)
1060 CALL HCHAR(5,CARROTS+5,
142)
1070 IF SP>0 THEN CALL SAY(,
LA$,,LA$,,LA$)
1080 IF CARROTS=0 THEN 1250
1090 GOTO 1140
1100 NEXT L
1110 FOR L=R(RR,CC)-16 TO R(
RR,CC)STEP SKIP
1120 CALL LOCATE(#20,L,C(RR,
CC))
1130 NEXT L :: IF DD<3 THEN
1140 ELSE 1030
1140 NEXT I
1150 CALL TALK("ROUND "&CC$(
ROUND)&" IS OVER")
1160 DISPLAY AT(3,7):"ROUND
"&STR$(ROUND)&" IS OVER"
1170 CALL TALK("#GOOD WORK#"
)
1180 CALL TALK("YOU HAVE "&C
C$(CARROTS)&" LEFT")
1190 DISPLAY AT(3,3):"YOU HA
VE "&STR$(CARROTS)&" CARROTS
 LEFT"
1200 FOR I=1 TO 400 :: NEXT
I
1210 ROUND=ROUND+1
1220 IF TIME>6 THEN TIME=TIM
E-2
1230 IF SKIP<5 THEN SKIP=SKI
P+.5
1240 GOTO 700
1250 CALL TALK("THEY HAVE WO
N")
1260 DISPLAY AT(3,1):" THE M
OLES ATE YOUR CARROTS"
1270 CALL TALK("#TRY AGAIN#"
)
1280 FOR I=1 TO 200 :: NEXT
I
1290 DISPLAY AT(3,1):" PRESS
 <ENTER> TO TRY AGAIN"
1300 RESTORE 1830
1310 READ LE,N1,N2
1320 IF LE=0 THEN 1300
1330 CALL SOUND(LE,N2,0,N2*2
,2,N1,4)
1340 CALL KEY(0,K,S):: IF S=
0 THEN 1310
1350 DISPLAY AT(3,1):""
1360 IF K<>13 THEN CALL TALK
("GOODBYE"):: CALL CLEAR ::
END ELSE 430
1370 CALL CHAR(112,"F090D050
50505F405F50505050D090F0")
1380 CALL CHAR(114,"3C242C28
2828E808E8282828282C243C")
1390 CALL CHAR(116,"F090D050
505050505050505050DF80FF")
1400 CALL CHAR(118,"00000000
00000000000000001CF404FC")
1410 CALL CHAR(120,"FF80DF50
50535E405E53505050DF80FF")
1420 CALL CHAR(122,"FC04F41C
008080808080000001CF404FC")
1430 CALL CHAR(124,"E0A05050
282815150A0A0505050D080F")
1440 CALL CHAR(126,"38285050
A0A0404080800000000808080")
1450 CALL CHAR(40,"704848545
45B5B5453350505050D090F0")
1460 CALL CHAR(42,"384848A8A
86868A8282828282C243C")
1470 CALL CHAR(44,"FF0000000
00000FF8181818181818181")
1480 CALL CHAR(46,"010204081
020408080402010080402 01")
1490 A$="pr"&CHR$(136)&CHR$(
138)&"tvxz!~  (*"&CHR$(136)&
CHR$(138)&"tvxz!~"
1500 B$="qs"&CHR$(137)&CHR$(
139)
1510 B$=B$&"uwy{}"&CHR$(127)
&"  )+"&CHR$(137)&CHR$(139)&
```

"uwy{}"&CHR$(127)

```
1520 FOR I=1 TO 7 :: DISPLAY
 AT(I,1):"":"   ";A$:"   ";B
$ :: NEXT I
1530 DISPLAY AT(11,9):"THE N
EXT DAY"
1540 CALL HCHAR(6,4,46)
1550 CALL HCHAR(6,5,44,24)
1560 CALL HCHAR(6,29,47)
1570 CALL VCHAR(7,4,45,6)
1580 CALL VCHAR(7,29,45,6)
1590 CALL HCHAR(13,4,47)
1600 CALL HCHAR(13,5,44,24)
1610 CALL HCHAR(13,29,46)
1620 DISPLAY AT(20,7):"1985
JB FREEWARE"
1630 DISPLAY AT(23,1):" HOLD
 DOWN ANY KEY TO BEGIN"
1640 RESTORE 1830
1650 READ LE,N1,N2
1660 IF LE=0 THEN 1640
1670 CALL KEY(0,K,S):: IF S<
>0 THEN 1700
1680 CALL SOUND(LE,N2,0,N2*2
,2,N1,4)
1690 GOTO 1650
1700 RETURN
1710 CALL CHAR(45,"000000007
C",46,"0000000000006060")
1720 DISPLAY AT(1,1):"
  INSTRUCTIONS"
1730 DISPLAY AT(4,1):"USE TH
E JOYSTICK TO MOVE THE"
1740 DISPLAY AT(6,1):"FARMER
 AROUND THE SCREEN."
1750 DISPLAY AT(8,1):"PRESS
THE FIRE BUTTON TO HIT"
1760 DISPLAY AT(10,1):"A MOL
E. IF YOU MISS - YOU"
1770 DISPLAY AT(12,1):"LOSE
A CARROT. WHEN THEY ARE"
1780 DISPLAY AT(14,1):"ALL G
ONE - THE GAME IS OVER."
1790 DISPLAY AT(18,1):" PRES
S FIRE BUTTON TO BEGIN"
1800 CALL KEY(1,K,S):: IF K<
>18 THEN 1800
1810 CALL CLEAR
1820 RETURN
1830 DATA 500,131,330
1840 DATA 500,30000,392
1850 DATA 500,131,262
1860 DATA 500,131,330
1870 DATA 250,196,330
1880 DATA 250,30000,294
1890 DATA 500,30000,349
1900 DATA 500,196,294
1910 DATA 250,30000,349
1920 DATA 250,30000,349
1930 DATA 500,196,247
1940 DATA 500,196,294
1950 DATA 250,131,294
1960 DATA 250,30000,262
1970 DATA 500,30000,330
1980 DATA 250,131,330
1990 DATA 250,30000,330
2000 DATA 250,30000,392
2010 DATA 250,30000,392
2020 DATA 500,131,262
2030 DATA 250,131,330
2040 DATA 250,131,330
2050 DATA 250,196,330
2060 DATA 250,30000,294
2070 DATA 500,30000,349
2080 DATA 500,196,294
2090 DATA 250,30000,349
2100 DATA 250,30000,349
2110 DATA 500,196,247
2120 DATA 500,196,294
2130 DATA 250,131,294
2140 DATA 250,30000,262
2150 DATA 900,30000,262
2160 DATA 100,30000,20000
2170 DATA 0,0,0
2180 SUB TALK(A$)
2190 CALL PEEK(-28672,A)
2200 IF A=0 THEN SUBEXIT
2210 CALL SAY(A$)
2220 SUBEND
```



```
1920 DATA 250,30000,349
```

BASICALLY YOURS Rich Klein

Hello again. I hope your holidays were happy ones. Now that they are over it's time to get those letters in. Send them to this column in care of the newsletter address found on the back cover of this issue.

This month we'll get back to Basics (pun intended). I've received several inquiries from several members and non-members about what VARIABLES are and what's the difference between the different types of variables. I will attempt to explain in this issue.

First, there are two basic types of variables that the 99/4A recognizes. One type is called a NUMERIC variable, while the other is called an ALPHANUMERIC or STRING variable. A numeric variable can handle only numbers and is represented by a name of some kind that begins with a letter and does NOT end in a Dollar sign ($). Variables that end in a Dollar sign represent alphanumeric or string variables. This is the distinguishing difference between the two types.

To make it clear, a numeric variable looks like: A, B, C1, APPLE, etc. Note that they start with a letter and do not end with a dollar sign. A numeric variable can be entered as a program statement in the following forms:

1) C=1
2) C=A+1
3) C=A+B
4) C=A*B

When a running program requests a numeric input, the response must be a number either in standard or scientific notation. It can't be a formula.

A string variable looks like this: A$, B$, C1$, APPLE$, etc. The same format of the variable names apply except that it must end in a dollar sign to tell the computer that it's not a numeric variable. A string, or alphanumeric variable can be any displayable or non-displayable character. Characters with an ASCII code from 0 to 255 can be part of a string variable. (More about ASCII later) Maximum length of string variables is 255 characters.

You might ask, "Why are variables necessary?". They are necessary because this is the only way you can keep track of numbers while they are in the computer. You can't see them; you can only refer to them. Actually, if you did not want the computer to manipulate the numbers for you, then you could work with them directly and you might produce a listing similar to Example 1 illustrated later on. But why did you buy the computer? To work the numbers for you. For that, you need to use variables.

The reason for two different types of variables is the way they are stored and used internally by the computer. Numeric variables are stored in a special way to allow for mathematical manipulation while string (we will refer to alphanumeric variables in this "generic" form to preserve simplicity) variables are stored "as is" in memory. We will discuss numeric variables first.

Single digit numbers or most numeric variables (A, APPLES, etc.)we

would use, use more space in memory as numeric variables than they would when expressed as a string variable. Each character of a string variable uses one byte of memory while a number uses eight bytes. Remember, the console has about 16,000 bytes of memory. These eight bytes representing the number are arranged in what's called RADIX 100 notation. This means the first byte contains a code which tells the computer what the place value of the next byte is (10,000,000,000;100,000,000.....00000001;0000000001). The next byte contains the number of units indicated by the first byte. For example, if the first byte says the next byte indicates 10,000's and the second byte has a 06 in it, it means that there are 6 ten thousands in the number or 60,000. Each succeeding byte is valued at the number contained times two decimal places less than the previous byte until all eight bytes are represented. If the number is, for example, 60,000, then all the remaining bytes will be filled with zeros. A numeric value is always expressed using eight bytes. Using this notation, full accuracy can be maintained to fourteen digits and numbers can be expressed in a range that spans -1E-128 to 9.99999999999999E+127 thru zero thru 1E-128 to 9.99999999999999E+127. These numbers are expressed and displayed accurately to 14 places or in scientific notation depending on the size of the number.

Complicated, isn't it? Well it isn't necessary to know all of this although it might be helpful to have an idea of how it works. Something to note is that the TI 99/4 and 4A are generally 6 or more decimal places MORE accurate than almost ALL the personal and home computers that have been on or are on the market today. This is both an asset and a liability to the TI. It is an asset because it allows for less error (almost negligible) than others in complicated formulas. It is a liability because to maintain full accuracy (even when not needed) the price you pay is time. It takes longer to do a simple formula than some of the others. Accuracy costs time.

A string variable (A$, APPLE$, etc.) is stored as a string of characters represented in memory as the ASCII code for each character of the string. If you look on your reference card for Basic or whatever, you will see somewhere on it a list of characters. Adjacent to each character you will see a number. This number is the ASCII code for that character. ASCII is the standard for encoding characters in the computer industry. A string variable uses as much space in memory as it is long plus one to indicate it's length to the computer. The maximum length for a string variable is 255 characters.

Numeric operators: +,-,*,/,^; can be used only with numeric variables.

String operator: &; can only be used with string variables.

Relational operators: >,<,=,>=,<=,<>; can be used with both types of variables, generally in IF-THEN-ELSE statements.

Numeric variables can be converted into strings and string variables containing only numeric characters can be converted into numeric variables. To convert a numeric variable into a string, simply include the statement (using A,A$ for example): A$=STR$(A). To convert a numeric string to a numeric variable simply use the statement: A=VAL(A$). If you attempt to convert improper variables, the program will crash with an error message.

If your'e wondering about variables used in arrays, they are the same except that they are modified to permit special handling that might not otherwise be possible with standard variables. I went into them to a small degree in answer to a letter in a previous column. If you require more info about them, send me a note.

As mentioned earlier, here is Example 1 and next to it is an example of the same thing when you let the computer do some of the work.

EXAMPLE 1:

```
100 PRINT 1+1
110 PRINT 1+2
120 PRINT 1+3
130 PRINT 1+4
140 PRINT 1+5
150 PRINT 1+6
160 etc...
32000 PRINT 1+300000
```

EXAMPLE 2:

```
100 A=1
110 A=A+1
120 PRINT A
130 GOTO 110
```

Looking at the examples presented (simple though they were), you should have at least a rudimentary idea of why a variable is necessary. In example 1, no variables were used and programming was tedious with no flexibility. In example 2, the variable A was used which made for a much shorter program and increased flexibility. It would be much easier to change program 2 to increment by two than program 1 (one line in 2 as opposed to all of them in 1).

I hope this article answers some of your questions about variables. If not, or if you have any questions, please write me at the group mailbox.

See you at the next meeting...

TRADING TImes:

Items for sale this month include a banner program written by 11 yr old computer wiz Steven Sherman. He is selling this program for $10 and you can reach him at 929-7620. Also up for sale by James Skellion are the following items: TI 99/4A console (new), fully loaded new expansion system including disk controller and drive, 32K card, RS232 and a new Multiplan. The price on this system is $700 or best offer and you can reach James at (312) 563-0217. Douglas Lee has for sale the following: PHP 1250 Disk Memory Drive, PHP 1240 Disk Memory System (controller and disk manager) for $200. He also has the Dynamic Data Direct Writer Tool for $40. You can call him at 534-6672 or wirte him at 811 Union Dr., University Pk, Ill 60466.

FORTH CHAPTER: Seaborn Smith

FLOATING POINT EXAMPLES

The following was handed out at the fair to show examples of using TI Forth Floating Point Routines. Create the following screens and study Chapter 7 in the TI Forth Manual to understand the Floating Point words used in the examples.

Interest Calculations

```
o Future Value    (FV) ---¦
o Interest Total  (IT)    ¦
o Interest Rate   (IR)    ¦--- Any three or four know values can
o Present Value   (PV)    ¦    solve the forth or fifth variable.
o Time Periods    (TP) ---¦
```

Formulas

Formulas in BASIC

```
                    TP
FV=PV(1+IR) FV=PV*(1+IR)^TP TP IT=PV((1+IR) -1) IT=PV*(((1+IR)^TP))-1 1/TP
IR=((FV/PV) -1) IR=((FV/PV)^(1/TP))-1 -TP PV=FV(1+IR) PV=FV*(1+IR)^(-TP)
ln(FV/PV) TP=---------- TP=(LOG(FV/PV))/(LOG(1+IR)) ln(1+IR) 0 ( Floating Point
Examples ) -FLOAT 1 0 VARIABLE fv 6 ALLOT 0 VARIABLE it 6 ALLOT 2 0 VARIABLE ir
6 ALLOT 0 VARIABLE pv 6 ALLOT 3 0 VARIABLE tp 6 ALLOT 16 SYSTEM 0 4 GOTOXY 4 :
FVT ir F@ >F 1 F+ tp F@ ^ pv F@ F* fv F! ; 5 : ITT ir F@ >F 1 F+ tp F@ ^
>F 1 F- pv F@ F* it F! ; 6 : IRT fv F@ pv F@ F/ >F 1 tp F@ F/ ^ >F 1 F- ir
F! ; 7 : PVT ir F@ >F 1 F+ >F -1 tp F@ F* ^ fv F@ F* pv F! ; 8 : TPT fv F@
pv F@ F/ LOG >F 1 ir F@ F+ LOG F/ tp F! ; 9 10 : DSP 0 13 GOTOXY ; 11 : S12
12 SPACES ; 12 : S20 20 SPACES ; 13 14 : ANS 9 12 GOTOXY F. S12 0 11 GOTOXY ;
15 : KIN PAD 12 BLANKS PAD 1+ 12 EXPECT VAL FAC> ; --> 0 ( Floating Point
Examples Cont. ) 1 : PVL CR ." Present Value? " KIN pv F! ; 2 : IRA CR ."
Interest Rate? " KIN ir F! ; 3 : TIM CR ." Time Periods ? " KIN tp F! ; 4 : FVL
CR ." Future Value ? " KIN fv F! ; 5 : FUTURE S20 DSP PVL IRA TIM FVT fv F@
ANS ; 6 : INTEREST S20 DSP PVL IRA TIM ITT it F@ ANS ; 7 : RATE S20 DSP FVL
PVL TIM IRT ir F@ ANS ; 8 : PRESENT S20 DSP FVL IRA TIM PVT pv F@ ANS ; 9 :
TIME S20 DSP FVL PVL IRA TPT tp F@ ANS ; 10 ." Key :" CR 11 ." FUTURE" CR 12
following describes how to use the FORTH SCREEN's to determine the future value
of $2000 for one year at 10% per month. FUTURE Present Value? 2000 ($2000.00)
Interest Rate? .1 (10% per month) Time Periods ? 12 (12 months) FUTURE
6276.856753 (IT=4276.8568)
```

966-2342: Irwin Goldstein

It's been approximately 4 weeks since I last sat down here and wrote about the users group bulletin board service. In that short time an amazing amount of change has come over the system. It is some of these changes I wish to talk about at this time.

We have added the convenience of chosing your preferred color scheme for both the background and letter shades of the screen. Once selected these colors will be remembered and stored as yours and will reappear each time you enter your password into the system. Adding this feature has saved me alot of listening to complaints about going blind and getting sick and so on and so on.

We have added SUB BOARDS to the basic board which provided a wealth of valuable information to users. We have listed for your convenience over 100 OTHER BOARDS within the Chicago Area, also we have provided a list of all the programs provided for you in the groups library, not to mention program reviews of some of the more popular third party software. Also listed is a complete rollcall of the members of the Board with their member numbers which makes it very easy to address your Electronic Mail.

I guess I should explain Electronic Mail. It is a message system that is made private and thus cannot be read by anyone other than the person it is addressed to. This is nice for those messages which could be embarrasing if made public. This is a convenience feature but should not be overused. Remember that if a message is private no one else can share in the information value of that message.

The electronic version of the group Newsletter has also be modified in that a menu has been added so the reader can now select his or her article he or she wishes to read while skipping over all the other articles. As it was before you were required to read the entire letter before you could read the last article. This menu also gives you the option of quiting the read before you start searching an article, thus saving valuable time.

Now to uploading, If you wish to submit a program for distribution over the download section we will gladly accept it. For your generosity, you will be rewarded with your choice of any program from the group library. Please bring any submissions to the next meeting or mail them to: THE CHICAGO TI USERS GROUP % BBS, P.O. BOX 578341, CHICAGO ILL, 60657. You will also be given credit for submitting the program. NO copyrighted material please. You can also try to reach us by (C)hat or leave us a message with your phone number and we will contact you.

Well that about does it for now. I hope you have been enjoying the new improved versions of the Board. If you have ideas for improvements or would like to see additional Sub Boards please let us know--leave a message addressed to sysop. Be talking to you and remembers you can usually reach me by going into chat. Someone will usually be around to talk. Bye for now and remember to call often.

VIDEO MODES: Jim Ness

THE TI99/4A
VIDEO DISPLAY PROCESSOR

Chapter 2

****

In the first chapter of this earth- shaking expose, I explained some of the ins and outs of the Grapphics and Text modes of the 9918A VDP processor (did I get the number right this time?).

Graphics mode gives you 24 columns of 32 characters each, with control of the screen and char colors (16 char sets can be seperate colors). In text mode, you get 24 columns of 40 chars each, with control of screen color and a single choice for char color.

In this chapter we move on to the oth- er two selections, Multi-color mode and Bit mode.

First though, we should talk a little bit about use of the VDP memory. VDP ram consists of 16k of memory, with the beg- inning used by the VDP for screen func- tions, and the rest used for temporary storage for peripherals, and, if you don't have memory expansion, for your Basic or Xbasic program. Each graphics mode uses the ram a little differently than each of the others.

In Graphics and Multicolor modes, the first 768 bytes are used for storing the characters currently at each screen pos- ition. In Text mode, the first 960 bytes perform this function, and in Bit Map mode, the second block, starting at byte 6144, and 768 bytes long, does this.

Other sections are the color table, which keeps track of char colors (not used in text mode, 32 bytes long in Graphics and Multicolor mode, and *6k* bytes long in Bit Map mode), pattern de- scriptor table, which keeps track of the shapes of the chars (2k bytes in all modes except Bit Mode, which uses 6k bytes for this function), and the sprite table (unused for Text mode, about 128 bytes in other modes). As you can see, Bit mode just about uses the entire 16k of VDP ram to store its information. As you can also see, it would take a *LOT* of work, many hours, to properly set up VDP ram for a Bit mode program.

*****

In Multicolor mode, you have control of 48 rows of 68 blocks, which can be any color from transparent to white, but can only be blocks, no special shapes. The idea is to build large multicolor shapes from these blocks (1/4 the size of the normal Graphics mode characters).

Sprites are available, also. A game like Munchman would be done in this mode.

*****

In Bit Map mode, you have control of 24 rows of 32 columns, **BUT** , each of the 768 boxes can be independently configures. You are not limited to just a couple hundred, as in Basic. Also, each box can have

as many as 16 colors attributed to it. In other words, the end result is that you have almost ul- timate control over each pixel on the screen. The only limitation is that in any 8-pixel row in any character, you can only have 2 colors. Not a big prob- lem.

What is a big problem, is, who has the time, just playing around, to write data configuring this much ram? We are talk- ing about 15k of ram, just to start the program. The other one k or so is left for you to use as you please. The time factor is why you see very little on the market with fancy Bit mode graphics. If a software house is going to put all this time into developing something, they have to know there will be a profit somewhere down the line.

*****

Well, that is it for this chapter. I had planned to finish it with 2, but in- stead, I will expand it to 3, with the final chapter explaining some of the programming necessary to use the VDP processor. -JN

LEARNING ASSEMBLY: Nick Iacovelli

```
*              PART 2
*
* Well part 2 is finally here. I was busy doing update to TI-WEST
* rather then writing this article
*
* My articles are now being reprinted in the user group newsletter.any
question * to me should be mailed to:

*   at      ARTICLES
*       c/o NICK IACOVELLI JR
*           1411 N. 36TH
*           MELROSE PARK IL 60160

* or should be left on the TI-WEST BBS at 312-766-2797 for quicker
* response All question will be printed and answered in the next
* article.
*
*
*
* In the last article I told you about the utilities and how to put a
character on the screen.  This article will show you how to put more
than 1 character on the screen move the character when the a key is
hit and fill up the screen with that character (call hchar)
*
*
*          LAST ARTICLE SHOW THIS
*
           DEF START
           REF VSBW
START      LI R0,400
           LI R1,>4100
           BLWP @VSBW
*
* This put a single character at screen location 400.
*
* Now let try filling up the screen with charcters.
```

```
          DEF START1
          REF VSBW
START1    LI R0,0
          LI R1,>4100
NEXT      BLWP @VSBW
          INC R0
          CI R0,768
          JNE NEXT
*          END
*
*
* The LI R0,0 set the screen location to 0.
* The INC R0 adds 1 to the screen location so the next location i want
* to write to is 1.
* The CI R0,768 check R0 for maximum screen location.(screen location
* is from 0 to 768)
* The JNE NEXT will tell the machine to goto NEXT if R0 does not equal
* 768 and will write the next character on the screen.JNE means jump
* not equal.
*
*
*
*
*       WRITING MORE THAN 1 CHARACTER
*              TO THE SCREEN
*
* The new command is the TEXT command this will allow you to write a
sentence * and display it on the screen.
*
*
*              THE PROGRAM
*
            DEF START2
            REF VMBW
T1   TEXT 'THIS WILL SHOW UP ON THE '
     TEXT 'SCREEN '
START2  LI R0,0     *SCREEN POSITION
        LI R1,T1    *LOCATION OF TEXT
        LI R2,34    *NUMBER OF CHARACTER
        BLWP @VMBW *DOES THE WRITE

*          END
*
*
*
*
*         THE USE OF THE KEYBOARD
*
* Just like the call key command in basic assembler works the same way
* with it BLWP @KSCAN Command.In order to map the keyboard in basic
* CALL KEY(#1,J,S) the #1 determines what keys are actives(more
* information is in your ti-basic book).In order to map the keyboard
* in assembler we must move a 1 to a location in memory. this locatio
* is >8374 the way to move to that location is.
*
*  LI R1,>0100
*  MOVB R1,@>8374
*
```

```
* The LI put the number we want in the memory. The MOVB (which stand
* for MOVE BYTE) move the first numbers to the location in the second
* half or loc 8374 in this case.
*
*        LETS MOVE ONE CHARACTER EVERY TIME WE HIT A KEY
*
*
       DEF START3
       REF VSBW,KSCAN
START3 LI R0,0       *SCREEN POSITION
       LI R1,>4100   *CHAR 'A'
       BLWP @VSBW
HITKEY LI R1,>0400    * MAP THE KEYBOARD
       MOVB R1,@>8374 *IN BASIC MODE
       BLWP @KSCAN
       LI R1,>FF00
       CB @>8375,R1 *CHECK TO SEE
       JEQ HITKEY    *WAS HIT NO GO BACK
       LI R1,>2000 *CHAR ' ' A SPACE
       BLWP @VSBW  *WRITE TO SCREEN
       INC  R0      *ADD 1 TO SCREEN POS
       CI R0,768    *LAST POSITION
       JEQ START3   *YES START OVER
       LI R1,>4100 *NO WRITE TO NEW POS
       BLWP @VSBW   *
       JMP HITKEY   *GOTO HITKEY
       END

*
*
* The 'C','CB','CI' are compare commands this command is use with the
* JMP,JNE, JEQ,JGT,JLT command.Think of these com mands as the IF THE
* ELSE statement in basic.
*    EXAMPLE
*       IN BASIC IT SHOWS LIKE THIS
* 5 REM
* 10 IF A=1 THEN 5 ELSE 11
*
*        IN ASSEMBLY
*
* TOP
*     C @A,@D1
*     JEQ TOP
*
* A and D1 would be labels to a data statement in assembly put in the
* assembly like this.
*
* A DATA >0001
* D1 DATA >0001
*
* You could change A or D1 By adding to it,subtracting from it or
* bye moving to it.I will continue this in my next article and
* explain most of * the instructions for moving data.
*
* In response to STEPHAN MEYERS article in the user group
* newsletter.NO FIRE STARTED BUT YOU MUST AGREE THAT FORTH DOES MAKE
* RATHER MESSY PIZZA AND THAT MAKES YOU PERFECT TO DELIVER IT * AS FOR
* ME ASSMEBLER IS A PIECE OF CAKE AND A LOT NEATER TO MAKE.
```

A FIFTH OF FORTH: Stephan Meyers

A few minor things to begin: I would like to thank all of you for your gracious support in the recent election, and to wish Sam Pincus luck in guiding our group for another term.
Last month there was a misprint in the Forth article, brought to my attention by some people at the last meeting. There was a line omitted on page 29, after the third complete paragraph. In the blank space beneath "While you are in the bottom part of the screen, type:" there should be the line: 0 0 127 255 LINE

Last month, after the meeting, those of us interested in Forth adjourned to Hank Ellerman's house to work a little bit, at which time I succeeded in confusing everybody terribly. Sorry, gang! Normally, we don't have acess to a machine, but by going over to Hank's house, we were able to use a three disk drive system to demonstrate some of the goings on in my article, without competing against Sam's class for time. If Hank is willing, we shall again move there after the regular meeting this month.

Incidentally, I must recommend the book STARTING FORTH by Leo Brodie of FORTH, Inc. It is a marvelous book, which explains most of these concepts far more elegantly than myself. I got mine at B. Dalton's for $17.95. Unfortunately, I just can't show things the way Mr. Brodie can, due to the limitations of the newsletter, as we shall see.

OK, class, settle down. Today we are going to talk about the stack. No lewd comments from you, Nick, about how stacked Elvira is. This is not the same sort of stack!

The TI Forth system has two stacks: The return stack and the value, or parameter stack. Both work about the same, but it is very dangerous to play with the return stack, so we don't. Thus, when I refer to the stack, I refer to the value stack, and when I mean return stack, I'll say return stack. The stack is sort of a temporary place to hold numbers. Stacks are used internally in most programming languages (Indeed, most programs), but are still hard to understand for most people, especially those who have not worked with an HP calclator. The stack is usually described as being like one of those tray dispensers at the cafeteria, the kind that you can take a tray and all the rest move up. In our analogy the trays are numbers, and, if you want to get technical, the trays don't move up, the countertop moves down. It's works the same way. In Basic, there is a return stack: When you do a GOSUB, the computer puts the line you came from on a stack. If you do several GOSUBs without a RETURN, the computer remembers them all up to the memory limits. When you do a RETURN, BASIC goes back to the last line GOSUBed from, always working with the top of the stack.

It would be nice if I could have pictures like in STARTING FORTH, but C'est la vie. Instead, I shall show the stack the way TI Forth does. The bottom is a vertical line to the left, like so: ¦ and the numbers trail off to the right. You normally can't see the stack, but there is a command on the disk called .S that shows you the stack. Now, let us assume that I have two numbers on the stack: A one and a two. This would look like so:

: 1 2

If you were to type 3 and then enter or space, FORTH would put this number on the top of the stack. Thus, the stack would look like this:

: 1 2 3

Forth will put any number or group of numbers on the stack, as long as they are seperated by spaces or carriage returns. Typing: 4 5 6 and enter will make the stack like so:
: 4 5 6

Simple, no? Forth can work on the stack with integer numbers from -32768 to +32767, or unsigned numbers from 0 to 65536. You can also use double precision arithmetic to allow some insane upper limit. Further, you can use floating point numbers (I.E. decimal) by loading the -FLOAT option. Forth actually doesn't understand floating point: Each floating point number is made up of four regular numbers, but the software allows Forth to deal with floating point. Floating point slows programs and wastes memory: It should be avoided at all costs. This is why I published the Forth CIRCLE routine some months back without using floating point. SIN and COS mess up your bit-map screen, and floating point is just too @#$@#$ slow!

Forth math operators work on this stack, making the Forth interpreter a reverse Polish notation system. Reverse Polish notation is called such because it was invented in Poland and because it does everything backwards. (Reader's of Truly Tasteless Jokes, by Blanche Knott, will say it figures). In Reverse Polish notation, the operator (I.E. +, -, *, /, etc.) follows both numbers that it acts on. In fact, everything in Forth works this way: Each word, or program, expects something on the stack and leaves a result. Thus, instead of the BASIC line:

```
IF 1+1=2 THEN PRINT "Of course it does!"
```

Would be in Forth:

```
1 1 + 2 = IF ." Of course it does!" ENDIF
```

Lets trace this, shall we? Typing 1 1 puts the number 1 on the stack twice, making it look like this: :1 1

Then the Forth interpreter executes +. + takes the top two numbers on the stack (ignoring everthing else, of course) and leaves their sum. Sort of like a machine. I fact, a whole Forth program can be seen as sort of like a series of machines. Numbers are sent through a series of machines. Each one processes the number and sends it to the next machine. Each machine ignores the numbers that aren't right on top, because they're none of it's business. Thus, the + machine takes the top two numbers, adds them up, adds them, and leaves the result. In this case, the stack would end up with: : 2

Then the 2 = part of our line would put a 2 on the top and compare the top two numbers. 2 would leave the stack like this: : 2 2

and executing the = "machine" would result in this: ! 1

This is because the Forth interpreter doesn't really understand Truth and Falsity (No computer does:C.F. Godel, Escher, Bach:An eternal golden braid, by Douglas Hofstader). 1 is the computer's way of saying "TRUE!". If the result were false, the stack would have a zero on top. In fact, BASIC does exactly the same thing. Try typing: PRINT 1+1=2

And you should get, I believe, a -1. You could also try:

IF 1 THEN PRINT "TRUE"

IF 0 THEN PRINT "FALSE"
This works in most BASICs, notably TI Basic. Any non-zero result is considered truth, and zero is considered falsity. One has to use a very high level language, usually a compiled one, to truly deal with truth and falsity. A good example of this is PASCAL.

Next in our line is IF. IF takes the top number on the stack, and like it's basic counterpart, it tests for a zero or non-zero value. Unlike BASIC, however, Forth puts the IF statement after the test. This is more of Forth's practicality. This way of doing things is not much harder to read than BASIC, once you learn it, and it is MUCH faster for the machine. If IF recieves a non-zero, or true, value, it executed the sequence of words after it until the ENDIF statement. You can say THEN instead of ENDIF, but I find this confusing, since it marks the end of the statement, whereas BASIC's THEN statement marks the beginning of things.

As I said above, Forth's way of doinf things is much faster than BASIC's. However, it is often clearer, once one is used to reverse polish notation. It is very easy to mix up things like: 1+(1*2)+2 and (1+1)*(2+2). They look alike, but are very different because 1+(1*2)+2=1+2+2=5 and (1+1)*(2+2)=2*4=6. In the words on Miss Emily Litella, "That's very different!" In Forth, we need no parentheses. The order of the operands leaves no ambiguity. These examples are a bit hard to read, but if we were working with variables instead of numbers, it would be very clear.

1+(1*2)+2 becomes 1 1 2 * 2 + + and

(1+1)*(2+2) becomes 1 1 + 2 2 + *

The logic follows the actual computation more closely than does the other notation. If you're curious, BASIC type notation is known as infix notation, and is notoriously hard for computers to understand. The BASIC interpreter contains a subroutine known as a parser to figure out just what it is you have in mind when you say: LET PROFIT=(INCOME-INCOME*.10)-EXPENSES

The final bit of our Forth line is: ." Of course it does!"
This is one of those things that I have to say: "Just use it and don't ask questions". It's one of the very complicated things in Forth to explain. I probably won't be able to really understand the ." word for a month or two, and I'd give you until next christmas to understand how it works on the inside. Unfortunately, I can't go very quickly in

this column. I have to go at a slow pace so any idiot can understand it, even Nick Iacovelli.

Until then, I'll tell you this much: ." works like PRINT in BASIC, only it can only be used for a string, such as

PRINT "HELLO!" becomes ." HELLO!"

Just remember, you can't put in any double quote marks (I.E. "") to get quote marks. You just can't get quote marks on the screen without doing tricky things. However, it is very easy to get Forth to print numbers. To have Forth print the top number in the stack, use a period (that is, .). '.' will type the top number on the stack. For example, typing:

```
69 .
```

Will put the number 69 on the stack, and then print it out to the screen.

Of course, Forth doesn't just use the stack for math! In fact, when you typed 0 0 127 255 LINE, you were putting those four numbers on the stack, then executing a routine that uses those four numbers as co-ordinates for drawing a line. There are also several operators just for manipulating the stack.

DUP duplicates the top number- ¦ 1 becomes ¦ 1 1

DROP drops the top number- ¦ 1 2 3 4 becomes ¦ 1 2 3

SWAP swaps the top two numbers- ¦ 1 2 becomes ¦ 2 1

ROT moves the third item to the top of the stack- ¦ 1 2 3 becomes ¦ 2 3 1

OVER copies the second item to the top- ¦ 1 2 becomes ¦ 1 2 1

(that one is a little weird, so read it carefully)

Well, that about wraps it up for this month. If I've done my job, you now know how to use the stack, how to manipulate numbers on the stack,how to do mathematics in Forth, and, most importantly, just what a stack is. Next month, I shall go into variables in Forth (unless the mood strikes me, in which case it shall be this month). After that, you'll learn how to write programs in Forth, how to load them, run them, etc. Hopefully, even Nick will understand it. Am I typing slow enough for you, Nicki? See, Nick's got a little problem: if he reads too long, his lips get tired. The other day, he left the keys in the car. He had to go find a coat hanger to let his sister out. Nick was going to be a pharmacist, but he couldn't figure out how to get the bottles into the typewriters. I got a million of 'em. Bye, now!!!

USING YOUR TI-WRITER: Roger M. Helm

COMMANDS

E EDIT - Typing or changing what you see on the screen.

T TABS - Used to set up tabulator settings and margins.

F FILES - What are you going to do with what is on the screen or on your disk.
  LF Loading a file you have on disk.
  SF Saving a file to the disk.
  PF Printing a file PIO or 5 30 PIO.

L LINES - Some things you can do with the lines on the screen.
  M MOVE - Moving lines to another place.
  C COPY - Duplicate any lines in another place.
  D DELETE - Gets rid of all the lines you select.
  S SHOW - Puts you at a line number you choose.

SH SEARCH
  FS FINDSTRING - Finding a word in the text.
  RS REPLACESTRING - Changing the spelling of all the same words.

SD SHOW DIRECTORY - Shows the names of files on your disk (like disk manager).
P PURGE - Clears out the text you have.

Q QUIT - Doesn't kill your text - yet.
  S SAVES
  P PURGES - Now it's dead if you didn't save it.
  E EXIT

OOPS - Puts back the line you just deleted!

Let's start ...

Power UP - Monitor on, PEB on, KEYBOARD on, PRINTER on, TI-WRITER Module in, TI-WRITER disk program in, and ANY KEY to start.

Select 2 for TI-WRITER then 1 for EDITOR. Disk drive loads program.
Put in initialized disk to store data (what you are typing).
Type "E" to edit or write.
Type "LF" to load a previous file. The name of file. (DSK1.TIWRITER)
Proceed to type material.
After 10 to 20 lines - hold down FCTN key and type "9" (COMMAND/ESCAPE).
Type "SF" then enter.
Type "DSK1.TIWRITER" then enter (this saves it under that name).
After saveing, the CURSOR will return to where you left typing and you can continue typing.

Other COMMANDs you may use:

BACK TAB	CTRL T
BEGINNING OF LINE	CTRL V
COMMAND/ESCAPE	CTRL C
DELETE CHARACTER	CTRL F / FCTN 9
DELETE END OF LINE	CTRL K

DELETE LINE	CTRL N / FCTN 3
DISPLAY LINE NUMBERS	FCTN 0 (ZERO)
DOWN ARROW/CURSOR	CTRL X / FCTN X
DUPLICATE LINE	CTRL 5
HOME CURSOR	CTRL L
INSERT BLANK LINE	CTRL O / FCTN 8
INSERT CHARACTER	CTRL G / FCTN 2
LAST PARAGRAPH/CURSOR	CTRL 6 / FCTN H
LEFT ARROW/CURSOR	CTRL S / FCTN S
LEFT MARGIN RELEASE	CTRL Y
NEW PAGE	CTRL 9 or P
NEW PARAGRAPH	CTRL 8 or M
NEXT PARAGRAPH/CURSOR	CTRL 4 or J
NEXT WINDOW-RIGHT	FCTN 5
OOPS!	CTRL 1 or Z
QUIT	FCTN =
REFORMAT	CTRL 2 or R
RIGHT ARROW/CURSOR	CTRL D / FCTN D
ROLL DOWN ONE PAGE	CTRL A / FCTN 4
ROLL UP ONE PAGE	CTRL B / FCTN 6
SCREEN COLOR	CTRL 3
UP ARROW/CURSOR	CTRL E / FCTN E
WORD TAB	CTRL 7 or W
WORD WRAP/FIXED MODE	CTRL O (ZERO)
SPECIAL CHAR ASCII	CTRL U

REMarks: Carole Goldstein

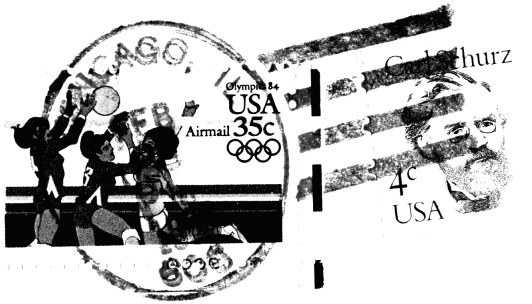
Some really exciting things are going on within the group lately. New executive board members were both elected and appointed and with the new people comes a rush of new ideas. The following is a list of officers for the current year:

SAM PINCUS-PRESIDENT
DAVE WAKELY-VICE PRESIDENT
HOWARD ROSENBERG-TREASURER
SANDY BARTELS-SECRETARY
LEN ROVNER-COMPTROLLER
CAROLE GOLDSTEIN-EDITOR/NEWSLETTER
RICH KLEIN-NEWSLETTER STAFF
BUTCH GOLDSTEIN-SYSOP GROUP BBS
CHUCK HOFF-HARDWARE CHAIRMAN
JON BARTELS-EVENTS CHAIRMAN
DAVE FOLEY-MEMBERSHIP CHAIRMAN
GRANT SCHMALGEMIER-ACQUISITIONS CHAIRMAN
KEN CZERWINSKI-TIRTON COLLEGE COORDINATOR
STEVE KAMP-HEAD GROUP LIBRARIAN
HANK ELLERMAN-ASSISTANT LIBRARIAN
JOHN BEHNKE-SERGEANT-AT-ARMS
BRIAN McMAHON-ASSISTANT SERGEANT-AT-ARMS

New advertising rates were set for this publication at the last executive committee meeting. Single issue ad prices remained the same at $10 for a half page ad and $20 for a full page. But now we will also be accepting multi-month package ads priced at $50 for a full page ad to run for three months and $25 for a half page ad to run three months. We hope that any companies dealing with TI99/4A products will

CHICAGO TI USER'S GROUP
PO BOX 578341
CHICAGO, IL 60657

FIRST CLASS MAIL
DATED MATERIAL
DO NOT DELAY



Tidings % Mr. Stephen Shaw

Englar

Magazine scanned 2021 by Stephen Shaw

---

use this newsletter to let our members know that they exist and that they have a product available for us. Anyone wanting to advertise should send camera ready copy and the appropriate fees to the group mailbox marked special attn newsletter.

Some of you may know that I have left my job at B. Dalton in Morton Grove. Those who do immediately want to know if that affects the group discount at that store. It should not. Your 10% group discount on computer book purchases still stands.

Well, that's about it this time around. Again thanks to all the people who have contributed their time and knowledge to help make our User's Group one of the best. See you February 9th.

ATTN: Check the color on your newsletter label. If it has a green dot this means that you have not paid your 1985 dues and you *will not* receive any more newsletters unitl dues are paid!